Float Support for Parameter Page

Page application

Mon, Aug 13, 2001

The little console parameter page application does not currently support floating point raw data channels. It makes requests for 16-bit raw data words only, and it uses the analog descriptor information to convert these values into engineering units. But a new feature has been added to the basic system code that supports raw floating point data, in which case the raw data is the engineering units floating point value, and no scaling is used at all. This feature uses a new FDATA table whose entries hold a reading, setting , nominal and tolerance values, all as 4-byte floating point values. The 2-byte fields in the ADATA table for these valus are not used for such channels, but the 2-byte alarm flags word and 2-byte trip count field *are* used. If the FLT bit (bit #12) in the alarm flags field is set, the channel uses raw floating point.

This note addresses the question of how to add support for the floating point raw data channels to the parameter page code. One can assume that some of the channels may be floating point channels, but not all. Given our history, it may easily be the case that no floating point channels are displayed on a given page. What this implies is that a separate data request made for such data might not be needed at all. In conformance with the current parameter page logic, readings and settings are acquired for each 15 Hz cycle, and nominal and tolerance values are collected every 13 cycles.

One simple approach is to read both integer and floating point values for each channel displayed. This will mean that much data is read that is unused, but the amount of data to be read, given the maximum of 14 channels, is not that large anyway.

A more complex approach would be to add more requests for collecting floating point data, only if it is needed by at least one of the displayed channels. Although implementing this approach may be a sign of personal virtue, it may not be worth doing. The data required is 112 bytes more of fast data and 112 bytes more of slow data. The totals would then be 168 bytes of fast data and 224 bytes of slow data.